Creating and Augmenting an xUCINET Data Set

These notes outline the process of moving from a raw data set to an xUCINET data file like the Zachary_KarateClub file that you inspected as part of the "Getting Started with xUCINET" notes. xUCINET data files are known as "data projects." They are known as "list objects" in the R lexicon.

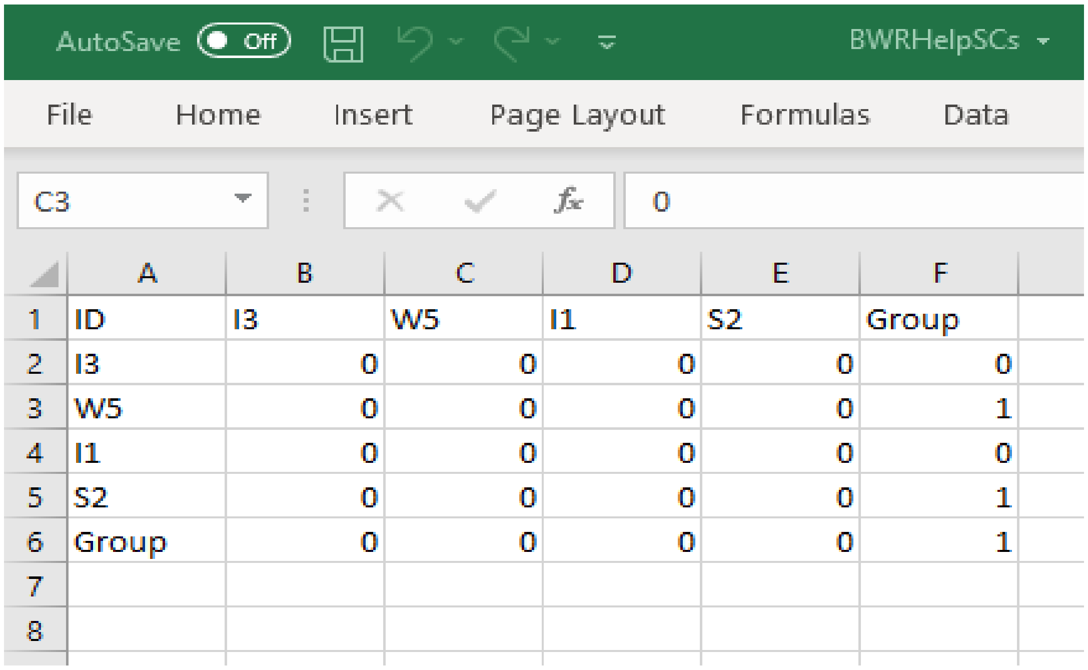
Data to be incorporated in a data project may reside in either a .csv ("comma separated values") text file, or in a previously-created R "matrix object." Here we will emphasize the .csv format, since you will ordinarily be creating data projects from data that do not already reside in R.

The minimal data project includes one network. It may be augmented by adding additional networks or attribute data. One begins with one network, and adds in the other information later on.

Preparing the data

First, you will need to create a .csv file to be imported into xUCINET. (We may do this step for you in exercises, but you will need to do it yourself when working with your own data.) This can be done using a spreadsheet editor like Microsoft Excel. Organize your data in the form of a data matrix; for a one-mode network, it will have an equal number of rows and columns. Labels for the nodes should appear in the top row and the leftmost column.

For the condensed digraph of the Bank Wiring Room helping network (slide 13 in the Directed Network session of class), a screenshot of the spreadsheet looks like this:



	A	B	C	D	E	F
1	ID	I3	W5	I1	S2	Group
2	I3	0	0	0	0	0
3	W5	0	0	0	0	1
4	I1	0	0	0	0	0
5	S2	0	0	0	0	1
6	Group	0	0	0	0	1
7						
8						

Save this as a comma separated values (.csv) file. That is a text file that looks like this:

```
ID,I3,W5,I1,S2,Group
I3,0,0,0,0,0
W5,0,0,0,0,1
I1,0,0,0,0,0
```

```
S2,0,0,0,0,1
Group,0,0,0,0,1
```

Creating the Data Project

Now start R and load package xUCINET via the `library()` command.

The R function for creating a data project is `xCreateProject()`. Its generic form is as follows:

```
xCreateProject(
  GeneralDescription = NULL,
  NetworkName = NULL,
  NETFILE1=,
  FileType = NULL,
  InFormatType = "AdjMat",
  NetworkDescription = NULL,
  Mode = NA,
  Directed = TRUE,
  Loops = FALSE,
  Values = NA,
  Class = "matrix",
  References)
```

To create a data project, you fill in necessary information about your project and your .csv data file by altering some of the default settings that appear to the right of "=" signs in this generic syntax. You will also assign the result produced by `xCreateProject()` to an R object. For this example, this could be the following (I've shown my modifications in red):

```
BWRHelp_Cond<-xCreateProject(
  GeneralDescription = "Condensed Digraph for BWR Helping Network",
  NetworkName = "HelpCondensed",
  NETFILE1=(file=file.choose()),
  FileType = "csv",
  InFormatType = "AdjMat",
  NetworkDescription = "Binary Connections among strong components",
  Mode = c("StrComps"),
  Directed = TRUE,
  Loops = TRUE,
  Values = "Binary",
  Class = "matrix",
  References="No references")
```

Commentary on the insertions shown in red:

```
GeneralDescription = "Condensed Digraph for BWR Helping Network",
```

Provides text information that describes the data project

```
NetworkName = "HelpCondensed",
```

Names the network that you are creating

`NETFILE1=(file=file.choose()),`

Provides the location where the network data are to be read from. Using `(file=file.choose()),` you navigate to the file while the function is being executed in R. If you know it, you can also insert the specific path to the data file (e.g. "Z:\COURSES\Courses\Soc2275\Data\BWRHelpSCs.csv") in place of `(file=file.choose())`.

`FileType = "csv",`

Tells R that you want it to take the data from a .csv file rather than an R object

```
NetworkDescription = "Binary Connections among strong
components",
```

Provides a brief description of the network

`Mode = c("StrComps"),`

An R list (i.e., `c()`) that tells the type of entity in the mode(s) that define the network. In this case the entities are strong components, which I have abbreviated as "StrComps"). The list contains only one entry, which means that you are defining a one-mode network; if it were to include two entries, you would be creating a project with a two-mode network

`Directed = TRUE,`

I did not adjust the default setting in this case, since the condensed network here is directed. If you are defining an undirected network, though, you should change this to `FALSE`

`Loops = TRUE,`

Says that diagonal elements are meaningful here. This is an unusual feature of this example, since it involves strong components that can be connected to themselves. Usually you will leave this at its default setting of `FALSE`

`Values = "Binary"`

Defines the level of measurement for the values in the network. In this case they are binary (0 or 1); they may also be `"Nominal"` (unordered, usually integers), `"Ordinal"` (ordered, usually integers), `"Rank"` (integers between 1 and some upper limit), `"Interval"` (quantitative, differences between values are meaningful), or `"Ratio"` (quantitative, ratios of values are meaningful)

`References="No references"`

Provides information on sources from which data were obtained

This yields an xUCINET data file called `BWRHelp_Cond` that looks like this:

```
> BWRHelp_Cond
$ProjectInfo
$ProjectInfo$GeneralDescription
[1] "Condensed Digraph for BWR Helping Network"

$ProjectInfo$Modes
[1] "StrComps"

$ProjectInfo$AttributesDescription
  Variable     Mode                              Details
1 NodeName StrComps Names of the nodes for mode StrComps

$ProjectInfo$NetworksDescription
    NetworkName                                  Details
1 HelpCondensed Binary Connections among strong components

$ProjectInfo$References
[1] "No references"


$Attributes
  NodeName
1       I3
2       W5
3       I1
4       S2
5    Group

$NetworkInfo
    NetworkName ModeSender ModeReceiver Directed Loops Values  Class
1 HelpCondensed   StrComps     StrComps     TRUE  TRUE Binary matrix

$HelpCondensed
      I3 W5 I1 S2 Group
I3     0  0  0  0     0
W5     0  0  0  0     1
I1     0  0  0  0     0
S2     0  0  0  0     1
Group  0  0  0  0     1
```

You can now work with this using xUCINET commands, referencing it as network
`BWRHelp_Cond$HelpCondensed.`

For example, its density is (note that I used the Loops= parameter here to indicate that the diagonal is valid in this special case)

```
> xDensity(BWRHelp_Cond$HelpCondensed,Loops=TRUE)

  .  ------------------------------------------------------------
```

```
 .   Number of valid cells: 25
 .   which corresponds to: 100 % of considered cells.
 .  ------------------------------------------------------------
```

```
[1] 0.12
```

Augmenting an existing data project by adding attribute data

One will often wish to supplement a data project with additional data about the nodes in the network. This can be accomplished via a process similar to that outlined above.

First, one creates a `.csv` file containing the attribute data. Continuing the example, I created a file that shows the size (Number of nodes) of each of the strong components, which looks like this:

```
ID,Size
I3,1
W5,1
I1,1
S2,1
Group,10
```

The identifiers for the nodes in the network should match those in the .csv file used to create the data project.

One then uses the `xAddAttributesToProject()` command to add the attribute data to the already-existing data project. The general syntax for that command is

```
xAddAttributesToProject(
  ProjectName,
  ATTFILE1 = NULL,
  FileType = NULL,
  AttributesDescription = NULL,
  Mode = NA
)
```

and I implement that here as follows:

```
BWRHelp_Cond<-xAddAttributesToProject(
  BWRHelp_Cond,
  ATTFILE1 = (file=file.choose()),
  FileType = "csv",
  AttributesDescription = c("Nnodes"),
  Mode = c("StrComps")
  )
```

We assign the output to the same object `(BWRHelp_Cond)` that we created before, thereby updating it.

Commentary on the insertions shown in red:

```
      BWRHelp_Cond,
```

Replace "Project Name" in the command's generic form with the name of the R object for the data project, here `BWRhelp_Cond`

```
ATTFILE1 = (file=file.choose()),
```

Either use `file=file.choose()` to navigate to the attribute data file as above, or locate and insert its path.

```
FileType = "csv",
```

Declare type of file for attribute data, either an existing `"Robject"` or a `.csv` file, as here

```
AttributesDescription = c("Nnodes"),
```

Give an R list (`c()`)including a short description or "variable label" describing each attribute. In this instance there is only one item in the list

```
Mode = c("StrComps")
```

An R list indicating the type of entity that the attributes describe. This should match a mode/entity type declared when the data project was created. It may seem redundant for a one-mode network, but with a two-mode network it is important to indicate which type of entity the attributes refer to.

After this function executes, we can inspect the contents of the data project to see that the attribute has been correctly added into `BWRHelp_Cond`:

```
> BWRHelp_Cond$Attributes
  NodeName Size
1       I3    1
2       W5    1
3       I1    1
4       S2    1
5    Group   10
```

(The attribute indeed has been added correctly. The rest of the data project file looks the same as shown above.)

Augmenting an existing data project by adding additional networks

If your project includes multiple networks, you can add additional networks in a similar manner, by way of the `xAddToProject()` command in xUCINET. We do not include details on that at this point, but may add something about it to these notes later on.